Lab 9

Sanchit Dass

# **Task 1: Generating two different files with the same MD5 hash**

This task involves the generation of two files with the same MD5 hash. For this task, we need to create a common prefix for both the files.

Generating the prefix:

A screenshot of a computer

Description automatically generated with medium confidence

Generating the distinct files using md5collgen tool:

Text

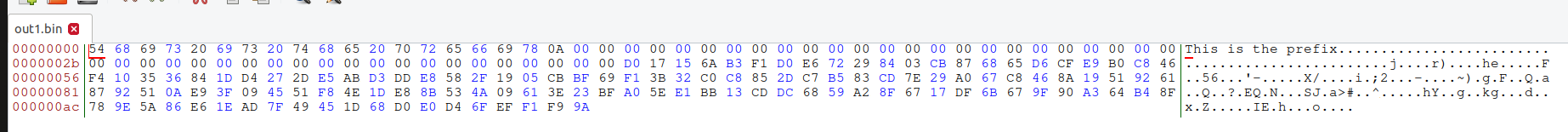
Description automatically generated

From the below screenshot we can see that the two files are different and have the same MD5 hash values:

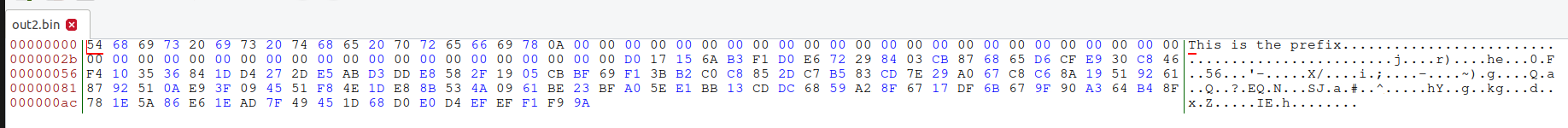
Text, chat or text message

Description automatically generated

Bless output of out1.bin:



Bless output of out2.bin:



From the above screenshots we can see that both the files have the same prefix but there is a slight difference in the rest of the hex bytes.

**Q1: If the length of the prefix is not multiple of 64, what is going to happen?**

**Ans:** From the example above, we can see that the length of the prefix is not a multiple of 64, it is 19 bytes.

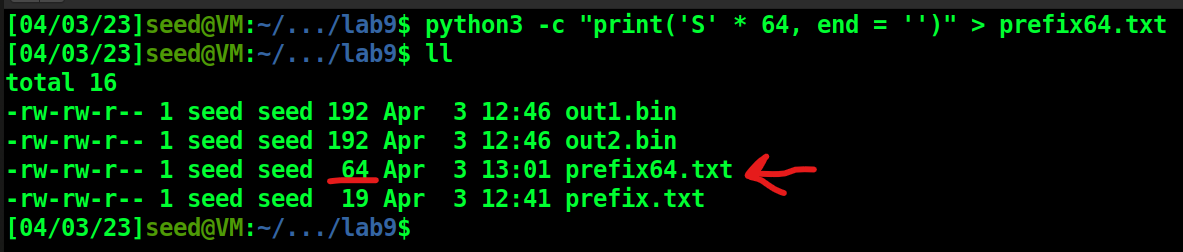
A screenshot of a computer

Description automatically generated with medium confidence

Since the md5collgen tool generates data of exactly 128 bytes, the total length of the prefix is bytes. This means that the tool padded the prefix to make it a multiple of 64.

**Q2: Create a prefix file with exactly 64 bytes, and run the collision tool again, and see what happens.**

**Ans:** Creating a prefix of length 64:



Running the tool again:

Text

Description automatically generated

The binary files are of length 192 again, which implies that the tool didn’t add any padding this time as the prefix was already a multiple of 64 bytes.

A screenshot of a computer

Description automatically generated with medium confidence

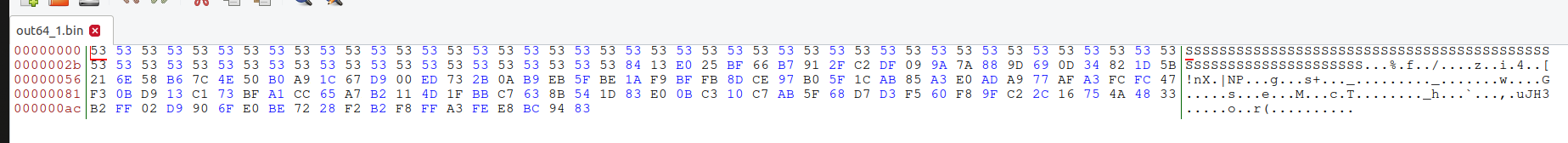
And again, we can see that the binary files are different but the MD5 hash values are the same:

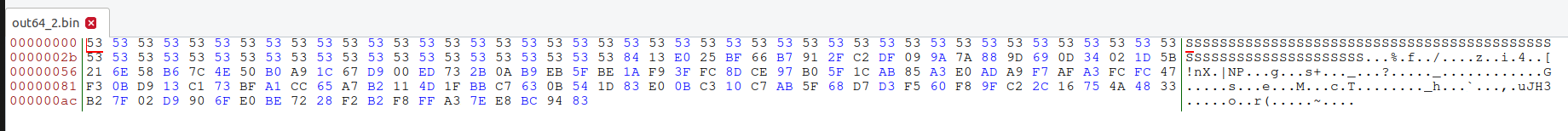
Text, application, chat or text message

Description automatically generated

**Q3: Are the data (128 bytes) generated by md5collgen completely different for the two output files? Please identify all the bytes that are different.**

**Ans**: The data generated by the tool are only slightly different:





I wrote a small python program to identify the different bytes in the two binary files:

*f1 = open('out64\_1.bin', 'rb')*

*f2 = open('out64\_2.bin', 'rb')*

*data1 = f1.read()*

*data2 = f2.read()*

*l1 = list()*

*l2 = list()*

*for item in data1:*

*l1.append(hex(item))*

*for item in data2:*

*l2.append(hex(item))*

*print("out64\_1.bin | out64\_2.bin")*

*print("------------------------------")*

*for i in range(len(l1)):*

*if l1[i] != l2[i]:*

*print(str(l1[i]) + "\t |\t " + str(l2[i]))*

*f1.close()*

*f2.close()*

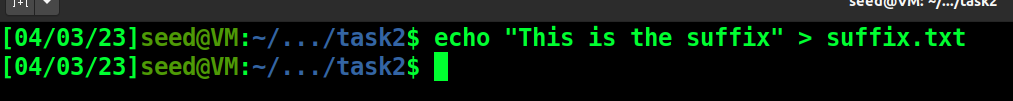
**Output of the program:**

Text

Description automatically generated

# **Task 2: Understanding MD5’s property**

Generating a suffix:



We can see that the binary files still have the same MD5 hash which demonstrated the property of length extension:

Text, chat or text message

Description automatically generated

# **Task 3: Generating two executable files with the same MD5 hash**

The program used as the starting point:

*#include <stdio.h>*

*unsigned char xyz[200] = {*

*0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,*

*0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,*

*0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,*

*0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,*

*0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,*

*0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,*

*0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,*

*0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,*

*0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,*

*0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,*

*0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,*

*0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,*

*0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,*

*0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,*

*0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,*

*0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,*

*0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,*

*0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,*

*0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41,*

*0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41*

*};*

*int main()*

*{*

*int i;*

*for (i=0; i<200; i++){*

*printf("%x", xyz[i]);*

*}*

*printf("\n");*

*}*

I created this script to generate the two binary files with the relevant prefixes and suffixes:

*#!/usr/bin/bash*

*gcc print\_array.c -Wall*

*head -c 12352 a.out > prefix*

*md5collgen -p prefix -o out1.bin out2.bin*

*echo -e "\n"*

*tail -c +12353 a.out > suffix*

*tail -c 128 out1.bin > P*

*tail -c 128 out2.bin > Q*

*cat prefix P suffix > a1.out*

*cat prefix Q suffix > a2.out*

*chmod 764 a1.out a2.out*

*echo -e "\nDiff of the final output files:"*

*diff a1.out a2.out*

*echo -e*

*echo -e "\nMD5 hashes of the output files:"*

*md5sum a1.out a2.out*

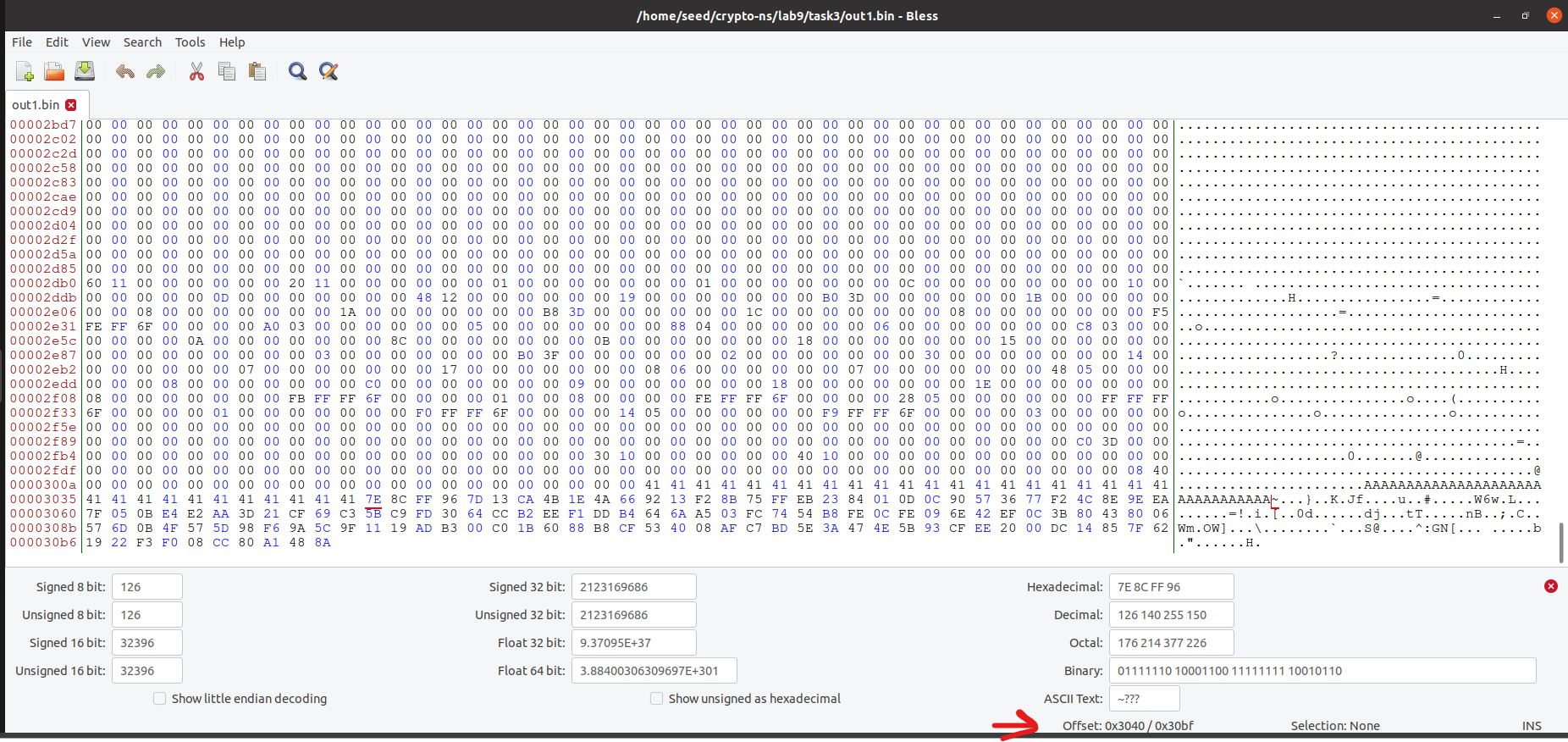
*echo -e "\nOutput of program 1\n"*

*./a1.out*

*echo -e "\nOutput of program 2\n"*

*./a2.out*

Bless output:



Output of the script:

Text

Description automatically generated

We can see that the data inserted by the md5collgen tool is in the outputs and it is slightly different as can be seen form the diff commands’ output. Also, the MD5 hashes are the same.

# **Task 4: Making the two programs behave differently**

This task involves the creation of two binaries which will make the same program behave differently. One will execute benign code and the other will execute malicious code.

Calendar

Description automatically generated

From the above screenshot we can see that the starting point of the first array is 0x3020. The nearest (least) address which is a multiple of 64 bytes is 0x3040 which is equal to 12352. We will use this address to calculate the prefix.

A picture containing text

Description automatically generatedFrom this screenshot we can see that the starting point of the second array is 0x31c0. The difference between the two starting points is 0x01a0 which is equal to 416. The length of suffix1 will be 416 – 128 = 288. Hence, the first 288 bytes of the suffix will be equal to suffix1. The second suffix will start from the 417th byte.

I created this script to execute all the instructions:

*#!/usr/bin/bash*

*# Compile the source code*

*gcc benign\_evil.c -Wall*

*# Extract the prefix*

*head -c 12352 a.out > prefix*

*# Create the collision files*

*md5collgen -p prefix -o out1.bin out2.bin*

*echo -e*

*# Check that they are different*

*diff out1.bin out2.bin*

*echo -e*

*# Check that they have the same hash*

*md5sum out1.bin out2.bin*

*# Extract the suffix*

*tail -c +12481 a.out > suffix*

*# Extract P and Q*

*tail -c 128 out1.bin > P*

*tail -c 128 out2.bin > Q*

*# Extract suffix1 and suffix2 from the suffix*

*head -c 288 suffix > suffix1*

*tail -c +417 suffix > suffix2*

*# Create the two binaries*

*cat prefix P suffix1 P suffix2 > a1.out*

*cat prefix Q suffix1 P suffix2 > a2.out*

*echo -e*

*chmod 764 a1.out a2.out*

*# Execute the binaries*

*./a1.out*

*echo -e*

*./a2.out*

*echo -e*

*md5sum a1.out a2.out*

The final output:

Text

Description automatically generated